Nonlinear Statistical Learning with Truncated Gaussian Graphical Models

Qinliang Su, Xuejun Liao, Changyou Chen, Lawrence Carin

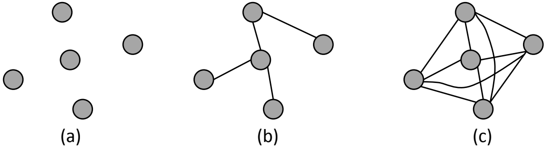
Department of Electrical & Computer Engineering, Duke University

Presented by: Qinliang Su

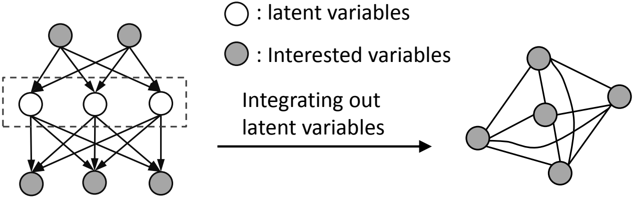
Jun. 20, 2016

Outline

- Graphical models encode statistical dependencies



- Dilemma: training easiness vs. modeling ability
- Solution: add latent variables to enhance modeling ability while maintaining simple graph structure



RBM and SBN are two good examples

Background (2)

- An important subclass: Gaussian graphical models (GGMs)
  - Many data can be well approximated by Gaussian
  - Admit efficient training due to Gaussian properties

- Limitations of GGMs
  - **(i)** Can only model Gaussian relations
  - **(ii)** Latent variables cannot enhance its modeling ability

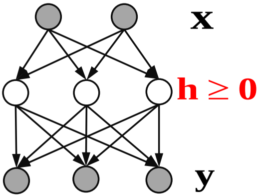
    No matter how many latent variables are added, the interested variables are always Gaussian distributed.

- Joint PDF

  Truncating the latent variables in GGM to be nonnegative

$$p(\mathbf{y},\mathbf{h}|\mathbf{x}) = \mathcal{N}_T(\mathbf{h}|\mathbf{W}_0\mathbf{x}+\mathbf{b}_0,\mathbf{P}_0^{-1}) \times \mathcal{N}(\mathbf{y}|\mathbf{W}_1\mathbf{h}+\mathbf{b}_1,\mathbf{P}_1^{-1}),$$

where $\mathcal{N}_T(\mathbf{x}\,|\boldsymbol{\mu},\mathbf{P}^{-1}) \triangleq \frac{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\mathbf{P}^{-1})\mathbb{I}(\mathbf{x}\geq\mathbf{0})}{\int_0^{\infty}\mathcal{N}(\mathbf{z}|\boldsymbol{\mu},\mathbf{P}^{-1})dz}$.



- Marginal PDF

$$p(\mathbf{y}\,|\mathbf{x}) = \underbrace{\mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_{\mathbf{y}|\mathbf{x}},\boldsymbol{\Sigma}_{\mathbf{y}|\mathbf{x}})}_{Gaussian} \underbrace{\frac{\int_0^{+\infty}\mathcal{N}(\mathbf{h}|\boldsymbol{\mu}_{\mathbf{h}|\mathbf{x},\mathbf{y}},\boldsymbol{\Sigma}_{\mathbf{h}|\mathbf{x},\mathbf{y}})d\mathbf{h}}{\int_0^{+\infty}\mathcal{N}\left(\mathbf{h}\,\middle|\,\mathbf{W}_0\mathbf{x}+\mathbf{b}_0,\mathbf{P}_0^{-1}\right)d\mathbf{h}}}_{Nonlinear\ \ modulation}$$

  Due to the nonlinear modulation, the distribution is no longer Gaussian

Truncated Gaussian Graphical Model (TGGM) (2)

- Visualizing the Output of TGGM

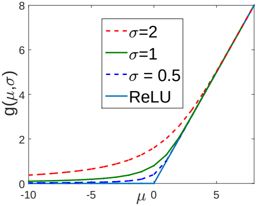
$$\mathbb{E}[\mathbf{y}|\mathbf{x}] = \mathbf{W}_1 \mathbb{E}[\mathbf{h}|\mathbf{x}] + \mathbf{b}_1$$

To understand the expression, if $\mathbf{P}_0 = \mathbf{P}_1 = \sigma^2 \mathbf{I}$, we have

$$\mathbb{E}[\mathbf{h}(k)|\mathbf{x}] = g\left(\mathbf{W}_0(k,:)\mathbf{x} + \mathbf{b}_0(k), \sigma\right),$$

where

$$g(\mu, \sigma) \triangleq \mu + \sigma \frac{\phi\left(\frac{\mu}{\sigma}\right)}{\Phi\left(\frac{\mu}{\sigma}\right)}$$



$g(\cdot)$ looks very similar to the ReLU nonlinearity in neural networks

- Advantages of TGGMs
  - **(i)** Inherit most properties of GGMs
  - **(ii)** Nonlinear modeling ability

Nonlinear Regression via TGGM (1)

- Modeling via TGGM

  Inspired by ReLU neural network, we model **X** and **Y** as

$$p(\mathbf{Y}, \mathbf{H}|\mathbf{X}; \boldsymbol{\Theta}) = \mathcal{N}_T(\mathbf{H}|\mathbf{W}_0\mathbf{X} + \mathbf{b}_0, \sigma_0^2\mathbf{I})\mathcal{N}(\mathbf{Y}|\mathbf{W}_1\mathbf{H} + \mathbf{b}_1, \sigma_1^2\mathbf{I})$$
$$= \frac{1}{Z(\mathbf{X}; \boldsymbol{\Theta})} e^{-E(\mathbf{Y},\mathbf{H}|\mathbf{X};\boldsymbol{\Theta})}$$

  where $E(\cdot) \triangleq \sum_{i=1}^{N} \frac{\|\mathbf{h}_i - \mathbf{W}_0\mathbf{x}_i\|^2}{2\sigma_0^2} + \sum_{i=1}^{N} \frac{\|\mathbf{y}_i - \mathbf{W}_1\mathbf{h}_i\|^2}{\sigma_1^2}$.

- Training via maximum-likelihood (ML)

$$\nabla_{\boldsymbol{\Theta}} \mathcal{Q} = -\mathbb{E}\left[\frac{\partial E}{\partial \boldsymbol{\Theta}}\middle| \mathbf{Y}, \mathbf{X}\right] + \mathbb{E}\left[\frac{\partial E}{\partial \boldsymbol{\Theta}}\middle| \mathbf{X}\right],$$

  By exploiting the properties of truncated normal and TGGMs, we have

  (i) $\mathbb{E}\left[\frac{\partial E}{\partial \boldsymbol{\Theta}}\middle|\mathbf{X}\right]$ can be computed in closed-form

  (ii) $\mathbb{E}\left[\frac{\partial E}{\partial \boldsymbol{\Theta}}\middle|\mathbf{Y}, \mathbf{X}\right]$ can be estimated using mean-field VB

Nonlinear Regression via TGGM (2)

- Training via backpropagation (BP)

$$\mathbb{E}[\mathbf{y}|\mathbf{x}] = \mathbf{W}_1 \mathbb{E}[\mathbf{h}|\mathbf{x}] + \mathbf{b}_1 \quad \text{with} \quad \mathbb{E}[\mathbf{h}(k)|\mathbf{x}] = g\left(\mathbf{W}_0(k,:)\mathbf{x} + \mathbf{b}_0(k), \sigma\right),$$

  - $\mathbb{E}[\mathbf{y}|\mathbf{x}]$ can be viewed as the output of a neural network with activation function $g(\cdot)$
  - Thus, it can be approximately trained using BP

- ML versus BP

  The updating equations of ML and BP are closely related, with only two differences

  (i) When updating $\mathbf{W}_1$, BP uses $\mathbb{E}[\mathbf{H}|\mathbf{X}]$, while ML uses $\mathbb{E}[\mathbf{H}|\mathbf{X}, \mathbf{Y}]$

  (ii) When updating $\mathbf{W}_0$, BP makes an incorrect Gaussian assumption

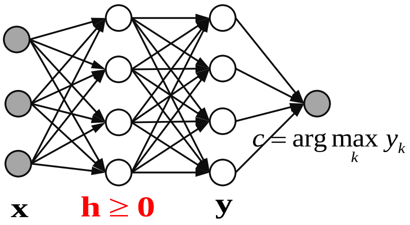
  ML is more efficient in exploiting data and more accurate in training, leading to better performance

- Classification

  We use *probit* model to transform the continuous Gaussian output to categorical output, i.e.,

$$p(c, \mathbf{y}, \mathbf{h}|\mathbf{x}; \boldsymbol{\Theta}) = \mathcal{N}_T(\mathbf{h}|\mathbf{W}_0\mathbf{x} + \mathbf{b}_0, \sigma_0^2\mathbf{I})\mathcal{N}(\mathbf{y}|\mathbf{W}_1\mathbf{h} + \mathbf{b}_1, \mathbf{I})$$
$$\times\, I\left(c = \arg\max_k y_k\right),$$

  where $c \in \{1, 2, \cdots, n\}$ is denoted as the n possible classes; $\mathbf{h}$ and $\mathbf{y}$ are latent variables.



Extensions to Other Learning Tasks (2)

- Re-representation as TGGM

  Define $\mathbf{z} = \mathbf{T}_c\mathbf{y}$, where $\mathbf{T}_c$ being a class-dependent matrix. Then, the input-output relation can be rewritten as

$$p(c, \mathbf{z}, \mathbf{h}|\mathbf{x}) = \mathcal{N}_T(\mathbf{h}|\mathbf{W}_0\mathbf{x} + \mathbf{b}_0, \sigma_0^2\mathbf{I})\mathcal{N}_T(\mathbf{z}|\mathbf{T}_c\left(\mathbf{W}_1\mathbf{h} + \mathbf{b}_1\right), \mathbf{T}_c\mathbf{T}_c^T)$$

  - Obviously, the above pdf can be represented by a TGGM
  - Thus, it can be trained similarly to its regression counterpart

- Deep models

  $\mathbf{P}_0$ is not necessary to be restricted to $\sigma_0^2\mathbf{I}$. As an example, by setting

$$p(\mathbf{h}|\mathbf{x}) \propto \exp\{-\frac{1}{2\sigma_0^2}\|\mathbf{h}^{(1)} - \mathbf{W}_0^{(1)}\mathbf{x} - \mathbf{b}_0^{(1)}\|^2\} \times \exp\{-\frac{1}{2\sigma_0^2}\|\mathbf{h}^{(2)} - \mathbf{W}_0^{(2)}\mathbf{h}^{(1)} - \mathbf{b}_0^{(2)}\|^2\}\mathbb{I}\left(\mathbf{h} \geq \mathbf{0}\right),$$

  we obtain a TGGM with two hidden layers, which can be trained similarly as previous models.

- Regression

  No. of hidden layer: 1;
  No. of hidden nodes: 100 for the two largest and 50 for the rest;

Table: Averaged Test RMSE and Std. Errors

Dataset	N	d	ReLU-BP	ReLU-PBP	TGGM-BP	TGGM-ML
Boston Housing	506	13	3.228±0.1951	3.014± 0.1800	2.927 ± 0.2910	**2.820± 0.2565**
Concrete Strength	1030	8	5.977± 0.0933	5.667± 0.0933	5.657 ± 0.2685	**5.395± 0.2404**
Energy Efficiency	768	8	1.098 ± 0.0738	1.804 ± 0.0481	**1.029 ± 0.1206**	1.244 ± 0.0979
Kin8nm	8192	8	0.091± 0.0015	0.098± 0.0007	0.088 ± 0.0025	**0.083 ± 0.0034**
Naval Propulsion	11934	16	0.001± 0.0001	0.006± 0.0000	**0.00057± 0.0001**	0.003 ± 0.0002
Cycle Power Plant	9568	4	4.182± 0.0402	4.124± 0.0345	**3.949 ± 0.1478**	4.183 ± 0.0955
Protein Structure	45730	9	4.539± 0.0288	4.732± 0.0130	4.477± 0.0483	**4.431 ± 0.0292**
Wine Quality Red	1599	11	0.645± 0.0098	0.635±0.0079	0.640 ± 0.0469	**0.625 ± 0.0340**
Yacht Hydrodynamic	308	6	1.182± 0.1645	1.015± 0.0542	0.957 ± 0.2319	**0.841 ± 0.2028**
Year Prediction MSD	515,345	90	8.932 ± N/A	**8.878 ± N/A**	8.918 ± N/A	9.002 ± N/A

- TGGM-BP generally performs better than ReLU neural networks
  - $g(\cdot)$ is more flexible than ReLU activation function for the extra σ^2;
  - The nonzero slop of $g(\cdot)$ as $\mu < 0$ makes optimization easier
- TGGM-ML performs best on most data sets
  - As analyzed previously, ML makes no incorrect assumptions and is more is efficient in exploiting data

- Classification

  One and two hidden layers are considered

Table: Test Accuracy of Classification

Methods	MNIST	20 News	Blog
ReLU (100)	97.58%	72.8%	65.86%
ReLU (200)	97.89%	73.27%	67.02%
ReLU (100-100)	97.83%	69.94%	67.93%
ReLU (200-200)	98.04%	69.91%	65.07%
TGGM-BP (100)	97.52%	73.65%	67.50%
TGGM-BP (200)	97.56%	73.62%	67.52%
TGGM-BP (100-100)	97.76%	71.06%	66.82%
TGGM-BP (200-200)	98.12%	71.18%	67.73%
TGGM-ML (100)	97.75%	**73.74%**	69.83%
TGGM-ML (200)	97.97%	73.38%	69.75%
TGGM-ML (100-100)	98.05%	68.01%	**69.89**%
TGGM-ML(200-200)	**98.31**%	67.52%	66.64%

TGGM-ML performs best on all data sets

Conclusions

- We proposed a nonlinear statistical learning framework with truncated Gaussian graphical model
- Nonlinear regression and classification tasks are cast into this framework by constructing appropriate TGGMs
- TGGMs can be further extended to deep models
- We show that all TGGM models can be trained efficiently by exploiting the properties of TGGM
- In the future, we will consider to further relax the structure of TGGM, e.g. lateral connection between hidden nodes; also, we will consider to use the model for unsupervised learning

Q&A